UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,612	12/02/2003	Kwasi Addo Asare	RSW9-2003-0192US1 (7161-1	5007

46320 7590 06/06/2007

CAREY, RODRIGUEZ, GREENBERG & PAUL, LLP
STEVEN M. GREENBERG
950 PENINSULA CORPORATE CIRCLE
SUITE 3020
BOCA RATON, FL 33487

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
06/06/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/725,612
Filing Date: December 02, 2003
Appellant(s): ASARE ET AL.

Scott D. Paul
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed March 29, 2007 appealing from the Office action mailed December 29, 2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The Examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,744,450	ZIMNIEWICZ et al.	6-2004
6,202,207	DONOHUE	3-2001

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 7 and 8 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 7 and 8 are directed to systems. However, the recited components of the systems appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. Therefore, these claim limitations can be reasonably interpreted as computer program modules—software *per se*. Furthermore, the specification discloses that the present invention can be realized in software *(see page 14, paragraph [0030])*. Therefore, the claims are directed to systems of functional descriptive material *per se*, and hence non-statutory.

The claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer

program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claims 1, 2, 4, 6-8, 10, 11, 13, and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zimniewicz et al. (US 6,744,450)** in view of **Donohue (US 6,202,207)**.

As per **Claim 1**, Zimniewicz et al. disclose:

- identifying target platform requirements for installation of a subject application component within a target specific installation script *(see Column 7: 6-9, "CD 81 and Disk 83 Information Managers provide required information concerning file location, size, etc. for the suite CD(s) during the integration process, and for the user's system onto which the suite will be installed."; Column 8: 18-25, "A baseline is a requirement determined by the suite owner. It concerns what OS/applications must be on a user's machine before installation of the suite can begin.")*;

- further identifying a listing of dependencies for said subject application and at least one specified relationship between said subject application and individual ones of said dependencies *(see Column 8: 53-54, "Suite baseline components may have dependent components." and 66-67 through Column 9: 1-5, "... consider a suite with Program A, Program B, and Program C included therein. Now assume that Program B is dependent on Program C,*

i.e. Program B needs Program C for proper operation. SIT will detect this dependency and will install Program C before Program B ..." and 18-21, "By default, SIT provides the Scenario Baseline page that displays what components need to be on the user's machine and what components are currently installed ...");

- enforcing both said target platform requirements and said at least one specified relationship prior to installing said subject application component *(see Figures 3 and 4a-4c; Column 7: 9-13, "During the installation process, the Setup Manager 79 utilizes the services of a Dependency Manager 85 to ensure that the required dependencies of the application programs within a suite are met." and 15-19, "A Validation Manager 87 is also used by the Setup Manager 79 to verify that required system components needed by the suite are met by the user's system for much the same reasons as for the Dependency Manager 85."; Column 8: 4-7, "The Dependency Manager 85 provided dependency checking and install-order calculation and verification across all selected components ..."; Column 9: 46-48, "Then, the Dependency Manager is called 96 to perform dependency checking among the components that user has selected."; Column 10: 15-19, "The suite installation requirements dynamic load libraries (dll) are then loaded 116 if specified, and a check is made 118 to determine if suite requirements have been met (i.e. RAM, processor, platform, pagefile, etc.)." and 23-24, "A check that the system meets the minimum requirements for the suite baseline is then performed 128.")*; and,

- aborting said installation where either one of said target platform requirements and said at least one specified relationship cannot be enforced *(see Column 8: 56-60, "If SIT cannot install the dependent components (and hence the suite baseline component), SIT informs the user, logs the error, terminates installation of the component which has a missing dependency,*

and continues with the rest of the installation."; Column 10: 24-27, "A check that the system meets the minimum requirements for the suite baseline is then performed 128. If the minimum requirements are not met, this is reported to the user 132. If the setup is unattended 134, the error is logged 136 and setup is terminated 126."), wherein said enforcing step comprises the steps of:

- determining whether all required ones of said dependencies can be accessed in said target platform *(see Column 11: 3-13, "... the component selection page presents the user with a tree view containing all of the components and their sub components. From this view the user may select or deselect a component and set its installation directory of any component.")*; and

- for each required one of said dependencies which cannot be accessed in said target platform, locating and installing said required one of said dependencies in said target platform *(see Column 9: 56-69, "The Setup Manager also installs 102, when necessary, components needed to achieve the scenario baseline if it differs from the suite baseline.")*.

However, Zimniewicz et al. do not disclose:

- updating dated ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies.

Donohue discloses:

- updating dated ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies *(see Column 9: 59-63, "The entries in the software updates list 60 include for each software product version 110 an identification 120 of the software resources required for applying the update and*

an identification 130 of its prerequisite software products and their version numbers."; Column 11: 46-63, "If all required resources are available locally (or on another machine in the case of software relying on some pre-requisite software operating on a remote machine), and have been verified, then the updater component progresses to the step 310 (see FIG. 4) of building the updated software version.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Donohue into the teaching of Zimniewicz et al. to include updating dated ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies. The modification would be obvious because one of ordinary skill in the art would be motivated to correct bugs and/or add new features in the software products *(see Donohue – Column 1: 21-23).*

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Zimniewicz et al. further disclose:

- wherein said further identifying step comprises the step of further identifying a listing of dependencies for said subject application and at least one specified relationship between said subject application and individual ones of said dependencies, wherein said at least one specified relationship is a relationship selected from the group consisting of a containment relationship, a usage relationship, a contradictory relationship and an equivalence relationship *(see Column 8: 66-67 through Column 9: 1-5, "... consider a suite with Program A, Program B, and Program C included therein. Now assume that Program B is dependent on Program C, i.e. Program B needs*

Program C for proper operation. SIT will detect this dependency and will install Program C before Program B ...").

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Zimniewicz et al. further disclose:

- wherein said determining step comprises the step of querying a registry of installed components in said target platform to identify components which have been installed in said target platform *(see Column 9: 33-36, "It then loads 86 the setup data file (setup.sdb) that contains general installation information, including scenarios, display order of components, list of startup and finish screens, etc."; Column 10: 30-32, "The suite baseline can also be specified in the setup data file ..." and 34-36, "Generally, suite baselines include OS, SPs, quick fix engineering or hot fixes (QFEs) (possibly as hidden components), Internet Explorer (IE), etc.").*

As per **Claim 6**, the rejection of **Claim 1** is incorporated; however, Zimniewicz et al. do not disclose:

- wherein said enforcing step further comprises the step of patching flawed ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies.

Donohue discloses:

- wherein said enforcing step further comprises the step of patching flawed ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies *(see Column 9: 67 through Column 10: 1-2,*

"In other cases, the resources comprise patch code for modifying an existing program (e.g. for error correction) and the patch's installation instructions."; Column 12: 13-19, "As examples, the software product to be updated may be a word processor application program. If the word processor as sold missed certain fonts or did not include a thesaurus, patches may subsequently be made available for adding these features. The updater component has the capability to add these to the word processor, subject to the update criteria.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Donohue into the teaching of Zimniewicz et al. to include wherein said enforcing step further comprises the step of patching flawed ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies. The modification would be obvious because one of ordinary skill in the art would be motivated to correct bugs and/or add new features in the software products *(see Donohue – Column 1: 21-23).*

As per **Claim 7**, Zimniewicz et al. disclose:

- a component installation engine configured to install application components and respective dependencies over a component distribution system *(see Figure 2: 79; Column 6: 40-44, "This system is embodied in a Suite Integration Toolkit (SIT) and utilizes a common architecture used for a setup database file (setup.sdb) to identify components and their available actions to be performed during the installation and setup thereof."; Column 7: 4-5, "The SIT includes a Setup Manager 79 that drives the installation process through the setup.sdb files for the suite.");*

- a script processor coupled to said engine and programmed to parse target specific installation scripts to identify both a listing of dependencies for the application components and at least one specified relationship between the application components and individual ones of said respective dependencies *(see Figure 2: 85; Column 7: 9-13, "During the installation process, the Setup Manager 79 utilizes the services of a Dependency Manager 85 to ensure that the required dependencies of the application programs within a suite are met.")*; and,

- a requirements verification processor programmed to enforce both target platform requirements for installing the application components and said at least one specified relationship prior to installing the application components *(see Figure 2: 85 and 87; Column 7: 15-19, "A Validation Manager 87 is also used by the Setup Manager 79 to verify that required system components needed by the suite are met by the user's system for much the same reasons as for the Dependency Manager 85.")*.

However, Zimniewicz et al. do not disclose:

- the component installation engine including a communicative coupling to a repository of updated ones of said dependencies; and

- wherein the component installation engine updates dated ones of said dependencies with said updated ones of said dependencies prior to installing said application components.

Donohue discloses:

- the component installation engine including a communicative coupling to a repository of updated ones of said dependencies *(see Column 8: 1-9, "The installation of each updater component includes the updater component registering itself with the operating system or another repository 40 on the local computer system. Thus, at least the updater components on*

the local system are identifiable and contactable by address information, and/or their product identifier, within the register entry. In alternative embodiments of the invention, the repository 40 may be a central or distributed repository for the network, as will be discussed below."); and

- wherein the component installation engine updates dated ones of said dependencies with said updated ones of said dependencies prior to installing said application components *(see Column 9: 59-63, "The entries in the software updates list 60 include for each software product version 110 an identification 120 of the software resources required for applying the update and an identification 130 of its prerequisite software products and their version numbers."; Column 11: 46-63, "If all required resources are available locally (or on another machine in the case of software relying on some pre-requisite software operating on a remote machine), and have been verified, then the updater component progresses to the step 310 (see FIG. 4) of building the updated software version.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Donohue into the teaching of Zimniewicz et al. to include wherein said enforcing step further comprises the step of patching flawed ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a central storage area of program modules for quick and efficient access and to correct bugs and/or add new features in the software products *(see Donohue – Column 1: 21-23)*.

As per **Claim 8**, the rejection of **Claim 7** is incorporated; and Zimniewicz et al. further disclose:

- wherein said at least one specified relationship comprises a relationship selected from the group consisting of a containment relationship, a usage relationship, a contradictory relationship and an equivalence relationship *(see Column 9: 3-4, "... Program B needs Program C for proper operation.")*.

Claims 10, 11, 13, and 15 are machine readable storage claims corresponding to the method claims above (Claims 1, 2, 4, and 6) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1, 2, 4, and 6.

(10) Response to Argument

a) **The rejection of Claims 7 and 8 under 35 U.S.C. § 101**

Appellant argues that the claimed invention is not directed to software *per se*, but instead, to a system *(see Appeal Brief – page 5)*. The Examiner maintains that the 35 U.S.C. § 101 rejections of Claims 7 and 8 are consistent with the Office's current policies regarding non-statutory subject matter. Appellant has submitted that the "claimed system must be coupled to physical components (i.e., hardware) to be functional *(see Appeal Brief – page 5)*." However, the recited components of the systems appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. Although the claims recite a script processor and a requirements verification processor, the Examiner has reasons to believe that the recited

processors of the systems can be reasonably interpreted as computer program modules—software *per se*, since the originally-filed specification provides no support for the processors as being hardware processors (*i.e.*, central processing units). Thus, absent an explicit and deliberate definition of what constitutes a processor, the claimed processors are interpreted as software *per se* under the broadest reasonable treatment. Furthermore, the originally-filed specification discloses that the present invention can be realized in software *(see page 14, paragraph [0030])*. Therefore, the claims are directed to functional descriptive material *per se*, and hence non-statutory.

b) The rejection of Claims 1, 2, 4, 6-8, 10, 11, 13, and 15 under 35 U.S.C. § 103 for obviousness based upon Zimniewicz in view of Donohue

Appellant argues that Donohue is silent with regard to when the updating occurs during the installation of a subject application component (emphasis in original) *(see Appeal Brief – page 5)*. Examiner would like to point out that Donohue is relied upon solely for the rejection of the particular claim limitation "updating dated ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies," which Donohue clearly discloses *(see Column 5: 54-62, "An updater component ... compares the available relevant updates with update criteria held on the local computer system ..., and then automatically downloads and applies software updates which satisfy the predefined criteria."; Column 11: 46-63, "The updater component performs 290 (see FIGS. 3 and 4) a scan of the operating system file system to check whether the required software*

resources are already available on the local computer system. The required resources are the software update artifacts required to bring the current application software to the new level, and the software updates required for updating prerequisite software to required levels." and "If all required resources are available locally ..., then the updater component progresses to the step 310 (see FIG. 4) of building the updated software version."). Thus, if the required software can be accessed in the local computer system, then updates to the pre-requisite software are performed to bring them to the required levels. Appellant has not presented arguments against this interpretation, but instead argues a feature that is not expressly recited in the particular claim limitation. Specifically, there is no requirement in the plain language of the particular claim limitation with regard to when the updating occurs.

However, the Examiner would like to respond to this argument by stating that Donohue clearly disclose that the updating occurs prior to the installation of a subject application component *(see Column 20: 3-17, "For all pre-requisites which are not already in place at the required revision level, an update is forced on their associated updater agents to make them migrate to the required level." and 65-67, "The updater component then implements the actual software upgrade: >>>>START Install_Resources()")*. Thus, prior to installing an upgrade of the software application, all pre-requisite programs of the software application must be updated to the required revision level.

Appellant further argues that the software in Donohue is not referred to as "software to be installed," but instead as installed software *(see Appeal Brief – page 7)*. Examiner disagrees. The "software to be installed" in Donohue is referred to as software updates (*e.g.*, fixes, additions,

and new versions). Donohue discloses an updater agent, which accesses relevant network locations and automatically downloads and **installs any available updates to its associated program** if those updates satisfy predefined update criteria of the updater agent (emphasis added) *(see Abstract)*. Furthermore, Appellant alleges that Donohue fails to teach or suggest the limitation of the enforcing step be performed "prior to installing said subject application component" (emphasis in original) *(see Appeal Brief – page 7)*. Examiner would like to again point out that Donohue is relied upon solely for the rejection of the particular claim limitation "updating dated ones of said required ones of said dependencies which can be accessed in said target platform with updated versions of said required ones of said dependencies" and not the particular limitation of the enforcing step be performed "prior to installing said subject application component," which Zimniewicz et al. clearly disclose *(see Figure 3; Column 8: 18-25, "A baseline is a requirement determined by the suite owner. It concerns what OS/applications must be on a user's machine before installation of the suite can begin."; Column 9: 51-63, "The Setup Manager also installs 102, when necessary, components needed to achieve the scenario baseline if it differs from the suite baseline."; Column 10: 15-19, "The suite installation requirements dynamic load libraries (dll) are then loaded 116 if specified, and a check is made 118 to determine if suite requirements have been met (i.e. RAM, processor, platform, pagefile, etc.).")*. Thus, before the installation of the suite can begin, a check is performed to make sure all the dependency requirements and hardware requirements are met. Appellant has not presented arguments against this interpretation.

Appellant further argues that Zimniewicz et al. is also silent with regard to when any updating occurs (emphasis in original) *(see Appeal Brief – page 7)*. Examiner disagrees. Zimniewicz et al. clearly disclose that the updating occurs prior to the installation of a subject application component *(see Figure 3; Column 7: 63-67 through Column 8: 1 and 2, "The installation process is actually divided into two primary stages: baseline and install."; Column 8: 18-25, "A baseline is a requirement determined by the suite owner. It concerns what OS/applications must be on a user's machine before installation of the suite can begin."; Column 9: 51-63, "The Setup Manager also installs 102, when necessary, components needed to achieve the scenario baseline if it differs from the suite baseline. This may occur, e.g., when a third party modifies a scenario data file, but neglects to modify the setup data file. As a result, the scenario baseline is no longer a subset of the suite baseline and requires the installation of additional components."; Column 10: 1-10, "... the data file shipped with a suite can be modified and shipped on a third party supplied CD ..." and "In this case, the Setup Manager will load the updated data file containing third-party product information as part of the startup.")*. Thus, prior to launching the installation stages, the data file is updated with modified third-party information and then loaded by the Setup Manger.

The Examiner would like to state that Zimniewicz et al. and Donohue are analogous art, since both inventions are in the same field of endeavor. Zimniewicz et al. teach a suite integration toolkit (SIT) used to install a suite of applications having multiple components and sub-components, where a Dependency Manager is utilized to ensure that the required dependencies of the application programs within a suite are met. Donohue teaches an updater

agent, which accesses relevant network locations and automatically downloads and installs any available updates to its associated program if those updates satisfy predefined update criteria of the updater agent, where any required pre-requisite programs are updated to maintain interoperability. Therefore, the rejections are proper based upon the combined teachings of Zimniewicz et al. and Donohue.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the Examiner in the Related Appeals and Interferences section of this Examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Qing Chen

Conferees:

Mary Steelman

Eddie Lee

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER